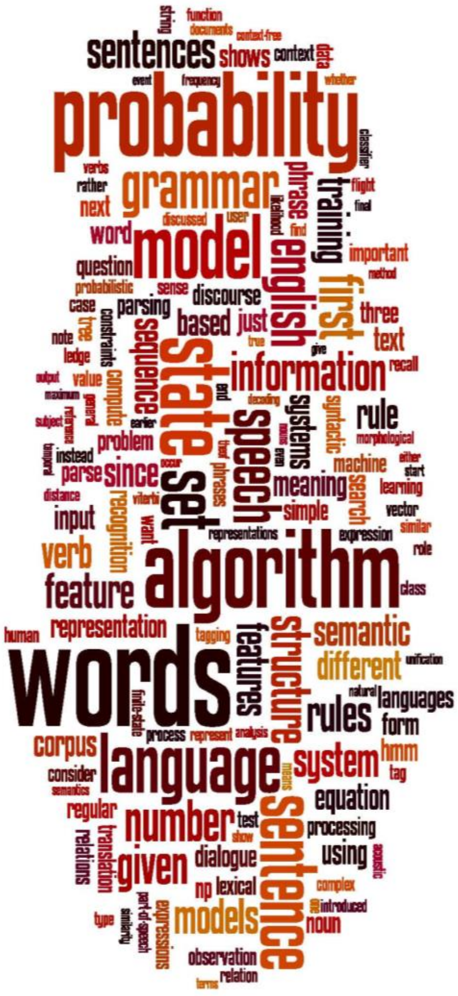


Basic Text Processing

Regular Expressions

Prof. Daniel Jurafsky, Stanford University

Presenter: Dr. Hamed Vahdat-Nejad

Regular Expressions: عبارات با قاعده

- Definition: An Algebraic notation for characterizing a set of strings
- They are useful for searching a pattern in text
- They are case sensitive

Regular Expressions: Disjunctions (گسست‌ها)

- Letters inside square brackets means any of them []

Pattern	Matches
`[wW]oodchuck`	Woodchuck, woodchuck
`[1234567890]`	Any digit

- Ranges `[A-Z]`

Pattern	Matches	
`[A-Z]`	An upper case letter	Drenched Blossoms
`[a-z]`	A lower case letter	my beans were impatient
`[0-9]`	A single digit	Chapter 1: Down the Rabbit Hole

Regular Expressions: Negation in Disjunction

- Negations ^
  - Carat means negation only when first in []

Pattern	Matches	
[^A-Z]	Not an upper case letter	
[^Ss]	Neither 'S' nor 's'	
[^e^]	Neither e nor ^	

Regular Expressions: More Disjunction

- Woodchucks is another name for groundhog!
- The pipe | for disjunction

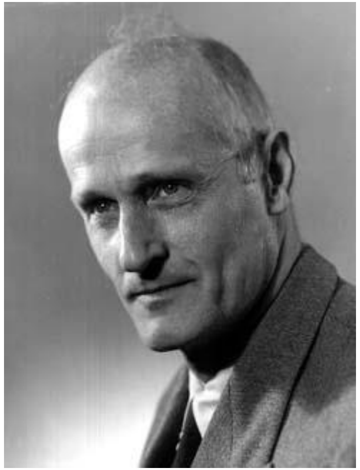
Pattern	Matches
`groundhog|woodchuck`	
`yours|mine`	`yours` `mine`
`a|b|c`	= [abc]
`[gG]roundhog|[Ww]oodchuck`	


Photo D. Fletcher

Dan Jurafsky

Regular Expressions: ? * + .

Pattern	Matches	
colou?r	Optional previous char	color colour
oo*h!	0 or more of previous char	oh! ooh! oooh! ooooh!
o+h!	1 or more of previous char	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n	Any character	begin begun begun beg3n



Stephen C Kleene

Kleene *, Kleene +

Summary

- Regular expressions play a surprisingly large role
  - Sophisticated sequences of regular expressions are often the first model for any text processing text
- For many hard tasks, we use machine learning classifiers
  - But regular expressions are used as features in the classifiers

Basic Text Processing

Word tokenization

Text Normalization

- Every NLP task needs to do text normalization:
  1. Segmenting/tokenizing words
  2. Normalizing word formats
  3. Segmenting sentences

How many words?

They picnicked by the pool, then lay back on the grass and looked at the stars.

- **Type**: an element of the vocabulary. Number of distinct words
- **Token**: an instance of that type in running text. Total number of words (Repetitions count)
- How many?
  - 16 tokens
  - 14 types

How many words?

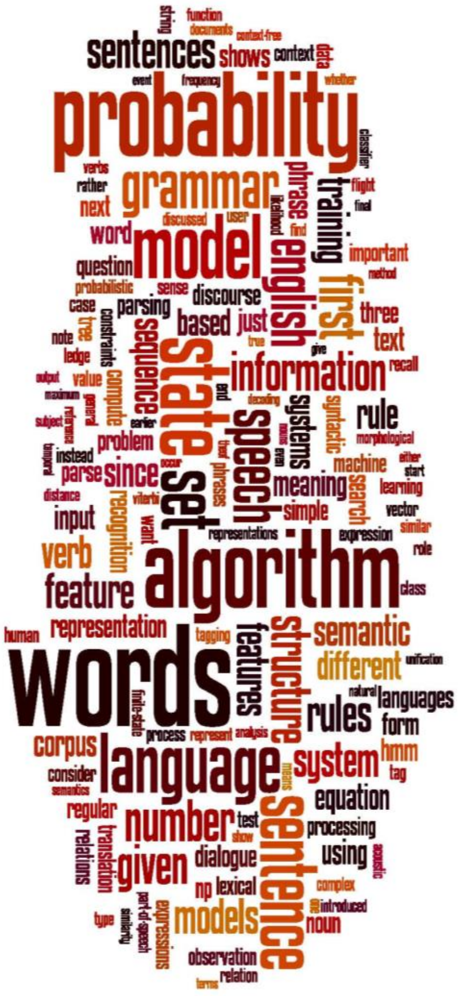
N = number of tokens

V = vocabulary = set of types

|*V*| is the size of the vocabulary

Church and Gale (1990): $|V| > O(N^{½})$

	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand



Basic Text Processing

Word Normalization and Stemming

Normalization

- Word normalization is the task of putting words/tokens in a standard format,

- Choosing a single normal form for words with multiple forms like USA and US or uh-huh and uhhuh.
  - We want to match ***U.S.A.*** and ***USA***

Case folding

- Case folding is a kind of normalization: Mapping everything to lower Case
- Suitable for Applications like Information retrieval: Since users tend to use lower case
- For sentiment analysis, Machine Translation, Information extraction
  - Case is helpful (***US (Country)*** versus ***us*** is important)

Lemmatization ریشه یابی

- Lemmatization is the task of determining that two words have the same root, despite their surface differences.
- Reduce variations to base form
  - *am, are, is* $\rightarrow$ *be*
  - *car, cars, car's, cars'* $\rightarrow$ *car*
- *the boy's cars are different colors* $\rightarrow$ *the boy car be different color*
- Lemmatization: have to find correct dictionary headword (سرواژه) form

Morphology

- The most sophisticated methods for lemmatization involve complete morphological (ریخت شناسی) parsing of the word.
- Morphology is the study of the way words are built up from smaller units called morphemes (واژک).
- Morphemes: The small meaningful units that make up words
  - **Stems (ساقه)** : the central morpheme, supplying the main meaning. (e.g. cat in cats)
  - **Affixes (ضمیمه)** : Bits and pieces that is added to stems. (e.g. s in cats)

Stemming

- Stemming is a simple and rudimentary algorithm for lemmatization.
- *Stemming* cuts affixes.
    - language dependent
    - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

for exampl compress and compress ar both accept as equival to compress

Porter's algorithm
The most common English stemmer

Step 1a

```
sses → ss   caresses → caress
ies  → i    ponies   → poni
ss   → ss   caress   → caress
s    → ø    cats     → cat
```

Step 1b

```
(*v*)ing → ø   walking   → walk
               sing      → sing
(*v*)ed  → ø   plastered → plaster
…
```

Step 2 (for long stems)

```
ational→ ate  relational→ relate
izer→ ize     digitizer → digitize
ator→ ate     operator  → operate
…
```

Step 3 (for longer stems)

```
al   → ø   revival    → reviv
able → ø   adjustable → adjust
ate  → ø   activate   → activ
…
```

Viewing morphology in a corpus
Why only strip –ing if there is a vowel?

(*v*)ing → ø	walking	→	walk
	sing	→	sing

Dealing with complex morphology is sometimes necessary

- Some languages requires complex morpheme segmentation
  - Turkish
  - Uygarlastiramadiklarimizdanmissinizcasina
  - `(behaving) as if you are among those whom we could not civilize'
  - Uygar `civilized' + las `become'

    + tir `cause' + ama `not able'

    + dik `past' + lar 'plural'

    + imiz 'p1pl' + dan 'abl'

    + mis 'past' + siniz '2pl' + casina 'as if'

Basic Text Processing

Sentence Segmentation

Sentence Segmentation

The most useful cues for segmenting a text into sentences are punctuation.

!, ? are relatively unambiguous

Period "." is quite ambiguous:

- Sentence boundary
- Abbreviations like Mr. or Dr.
- Numbers like .02% or 4.3

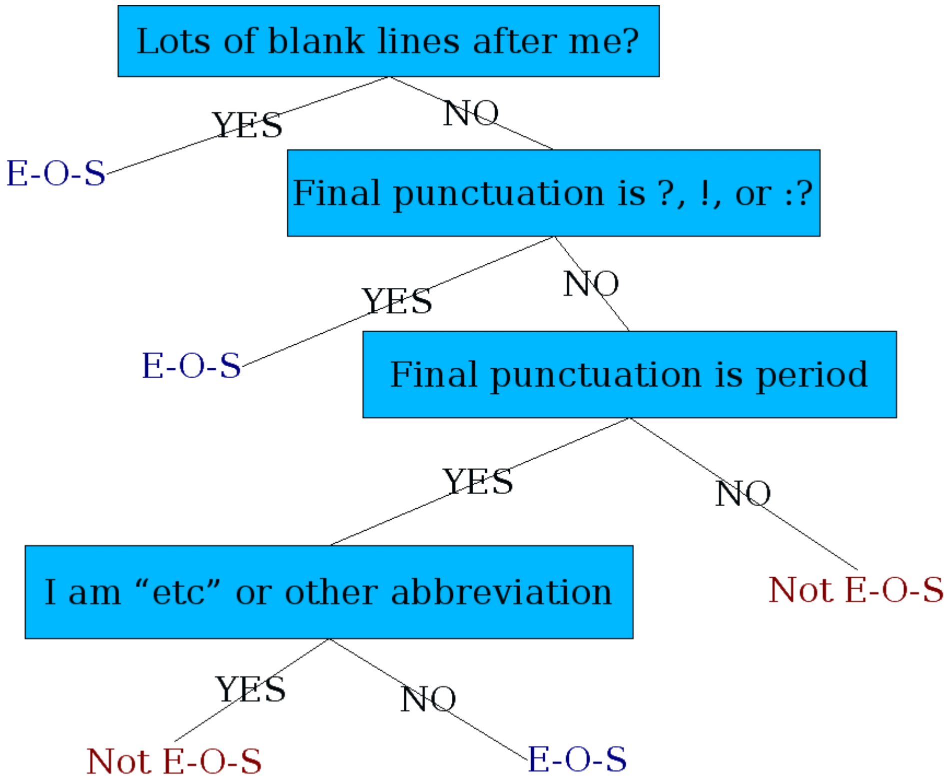
Sentence tokenization methods work by first deciding (based on rules or machine learning) whether a period is part of the word or is a sentence-boundary marker.

An abbreviation dictionary can help determine whether the period is part of a commonly used abbreviation;

Build a binary classifier that Looks at a "." and decides:

- Decides EndOfSentence/NotEndOfSentence

Determining if a word is end-of-sentence: a Decision Tree



More sophisticated decision tree features

- Case of word with ".": Upper, Lower, Cap, Number
- Case of word after ".": Upper, Lower, Cap, Number

- Numeric features
  - Length of word with "."
  - Probability(word with "." occurs at end-of-s)
  - Probability(word after "." occurs at beginning-of-s)

Implementing Decision Trees

- A decision tree is just an if-then-else statement
- The interesting research is choosing the features
- Setting up the structure is often too hard to do by hand
  - Hand-building only possible for very simple features, domains
    - For numeric features, it's too hard to pick each threshold
  - Instead, structure usually learned by machine learning from a training corpus

Decision Trees and other classifiers

- We can think of the questions in a decision tree
- As features that could be exploited by any kind of classifier
  - Logistic regression
  - SVM
  - Neural Nets
  - etc.